

UPFrame: UDP Processing Framework

An Extendible Framework for the Reception and Processing of UDP Data

ETH DDoSVax team and Caspar Schlegel
TIK, Communication Systems Group

Table of Contents

1	Introduction	1
1.1	UPFrame	1
1.2	Application Domains	1
2	Framework Components	3
2.1	Writer	3
2.2	Reader Processes	3
2.3	Mgmt Process	3
2.3.1	Free	4
2.3.2	Filled	4
2.3.3	Trashed	5
2.4	Monitor	5
2.4.1	Stat	5
2.4.2	Statd	5
2.5	Watchdog	5
3	Plugin Support	6
3.1	Concepts	6
3.2	TrafficShaper based	6
3.3	Without TrafficShaper	7
4	Installation	10
4.1	Prerequisites	10
4.2	Compilation	10
4.3	Configuration	11
4.3.1	Directory Layout	11
4.3.2	Environment Variables	11
4.3.3	Config Files	11
4.3.3.1	mgmt.cfg	11
4.3.3.2	writer.cfg	13
4.3.3.3	startpw.sh	13
4.4	Contributions	14
4.4.1	Statistics Viewer	14
4.4.2	Portwatch	14
4.4.3	Installation of the Web Scripts	14
4.4.4	Installation of the Portwatch Plugin	15
5	Operation	17
5.1	Starting	17
5.1.1	Multiple Instances	18
5.2	Statistics	18
5.2.1	Summary View -S	18
5.2.2	Dataflow View -D	18
5.2.3	Memory View -M	18
5.2.4	Receiver View -R	19
5.2.5	TrafficShaper View -T	19
5.3	Running the Watchdog	19

6	Guide to Plugin Implementation	21
6.1	Plugins without the TrafficShaper	21
6.2	Plugins using the TrafficShaper	24
	GNU GENERAL PUBLIC LICENSE	26
	Preamble	26
	TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION	26
	Appendix: How to Apply These Terms to Your New Programs	31

1 Introduction

1.1 UPFrame

UPFrame is an application framework released under the GNU public license that is able to:

- Receive and process incoming UDP packets at fast rates
- Buffer several megabytes of incoming data to smoothen out data bursts
- Feed the received packets to plugins that independently process the data in the packets

Its features are:

- The framework was designed to be fast, stable and resource efficient. In periods of low network load the memory footprint is minimal while the framework is able to buffer large amounts of data during busy times. The framework is therefore able to capture data of ongoing DDoS attacks.
- There are mechanisms that ensure the proper operation of the framework even in case of a malfunctioning plugin. If parts of the framework crash, there is an automatic data recovery mechanism that tries to save received but not yet processed data.
- The current operational state of the framework like buffer allocation, number of incoming packets etc. can be observed using a web interface.
- The available PortWatch plugin processes flow-level router data (received as UDP packets that contain Cisco Netflow v5 records) and calculates statistics about bandwidths of several well-known UDP, TCP and ICMP ports, as well as statistics of single ports.
- The framework runs on Linux (x86 and amd64) and FreeBSD (x86). Reports of succeeded or failed installation attempts are welcome. The footprint of the framework itself is below 1 MB, the size of the buffering component is configurable, its default size is 100 MB RAM. The CPU load imposed on the system by the framework is very low which allows for heavy and complex calculations performed by the plugins.

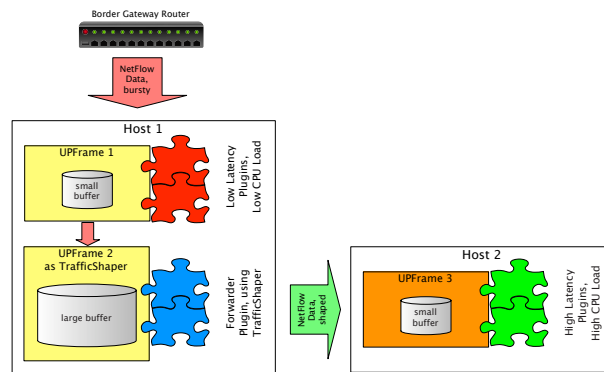
1.2 Application Domains

The primary application of the framework is the reception and processing of UDP data in near realtime.

- The DDoSVax¹ team at TIK, ETH Zurich uses the framework to implement and test new algorithms for the characterization of worm and DDoS traffic in an Internet backbone. Specifically, it is used to process Cisco NetFlow data in near realtime. The example Port-Watch plugin uses the NetFlow data to calculate statistics about the protocols and ports of the IP traffic flowing in and out of a specific network. Other plugins using the NetFlow data could, e.g., calculate the amount of traffic for accounting reasons.
- The framework can also be used as a UDP splitter or reflector which receives and resends UDP packets. There is a TrafficShaper facility that plugins can use, which flattens bursts of data (for example created by the periodical export of the NetFlow data from Cisco routers). The use of the TrafficShaper therefore reduces the load imposed on the network.
- Multiple instances of the framework can be cascaded so processes with different demands on latency can be served. In the example below the router is exporting its bursty NetFlow data to *UPFrame 1* on host 1. Its task is to serve the data to some low latency plugins that need the data in the very moment it is received. The data then is passed to *UPFrame 2*, which

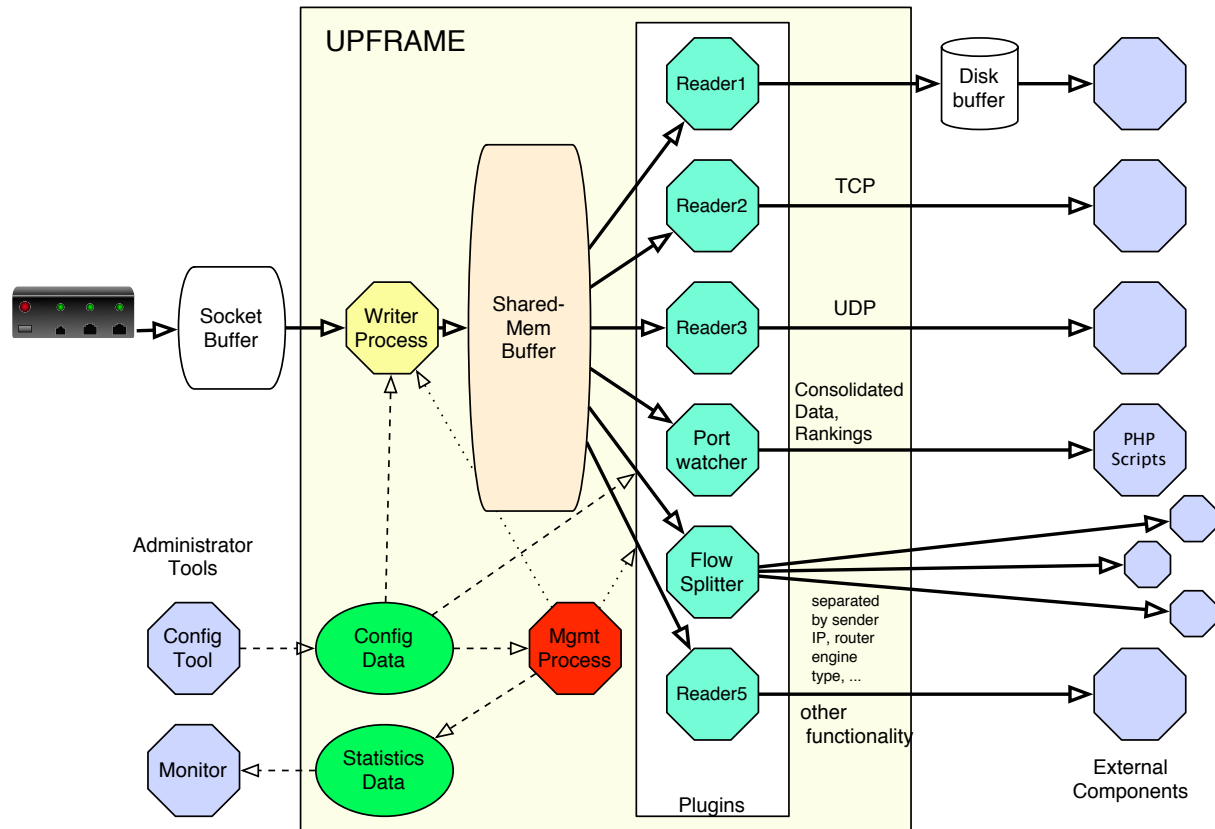
¹ <http://www.tik.ee.ethz.ch/~ddosvax/>

uses the TrafficShaper (and therefore a large buffer) to even out the data rate changes. The data is then forwarded to *UPFrame 3* on host 2 where the processing intensive plugins reside. These plugins have a low demand on the latency. With this layout, both types of plugins are served. The network load between host 1 and host 2 is sustained and regular.



2 Framework Components

The framework is highly configurable but for some of the configuration options it is important to understand the internal structures and algorithms of the framework.



The system consists of several processes working hand in hand.

2.1 Writer

The Writer Process is responsible for reading the received data from the network socket buffer. The data is fed into a memory segment (see the section about the mgmt process below) and injected into the buffer.

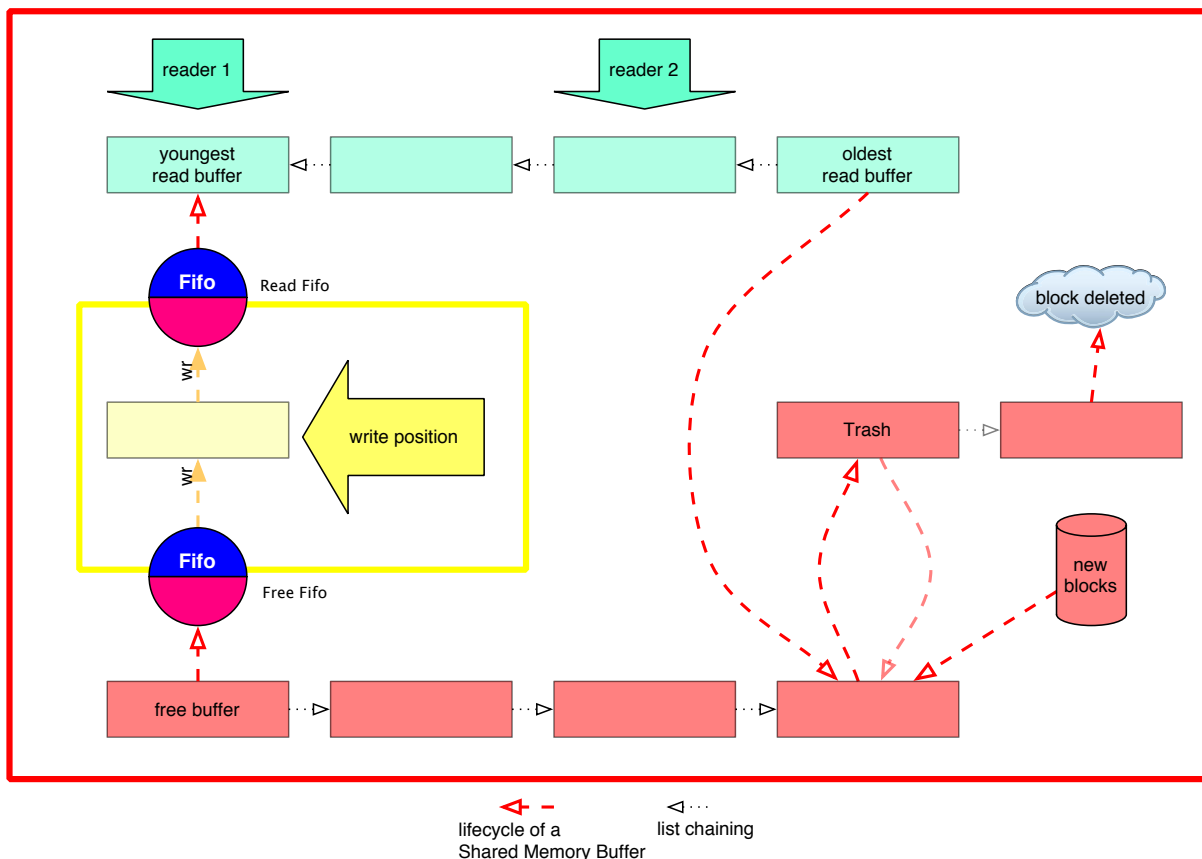
2.2 Reader Processes

The Reader Processes are the plugin modules of the framework. They can perform arbitrary calculations on the received data. The plugins are fully independent from each other and can process the data in their own speed as long as in average the processing speed is as fast as data is flowing in. Even in case a plugin is constantly too slow, the other plugins keep on receiving fresh data.

There is a library providing the writer of a plugin with functions to access the data in the buffer. See the section about the plugin library for further explanation.

2.3 Mgmt Process

The Memory Management Process' responsibility is to manage the shared memory buffer. The shared memory buffer consists of segments of System V Shared Memory of equal size.



This buffer is organized as shown in the illustration above. Shared Memory segments are circulating between the different components of the framework. There are several stations in the lifecycle of a shared memory segment. The internals of this lifecycle are important for configuring the sizes of the data structures described herein:

There are three states a segment can be in: free, filled and trashed.

2.3.1 Free

A newly allocated or recycled segment is enqueued in the free buffer. The free buffer is a repository of ready-to-use shared memory segments. The reason for keeping a stock of these is to even out the load on the machine the framework is running on: in case of a heavy inflow of data (for example an ongoing denial of service attack) the operating system must not waste time in allocating new Shared Memory segments but the framework can serve from ready to use segments from the free buffer. This lessens the probability of losing data in the writer process. This buffer has a certain minimal and maximal size that is ensured by the Memory Management Process and can be configured.

The Memory Management Process will enqueue a free segment in the free fifo. The free fifo is a decoupling element: as long as there are enough free segments in the free fifo, the free fifo guarantees a lock- and wait-free access of the writer on the memory segments. Again this lessens the probability of losing data because of socket buffer overflow.

2.3.2 Filled

After the Writer Process filled the Shared Memory Segment with received data, it is enqueued in a fifo: the read fifo. Again this is a decoupling element. The Memory Management Process moves the filled segment from the read fifo into the read list. In here it is processed by the reader plugins. As soon as all plugins have processed the data it is moved back into the free list and changes to state free. More on this topic can be read in the next chapter about the plugin concepts.

2.3.3 Trashed

To ensure a certain maximum memory footprint the free buffer must not exceed a certain length. As soon as this length is reached newly enqueued segments are moved to the trash list. After a certain time (or when the trash list exceeds a certain length) segments are deleted and the memory is given back to the operating system.

2.4 Monitor

Monitors can observe statistical data of the framework. At the time of writing of this document there exist two monitor programs:

2.4.1 Stat

The statd program is a program that dumps the statistical information in a textual representation ready to be fed into gnuplot.

2.4.2 Statd

Statd is a network server. Processes can connect to it via TCP and can collect the statistical data. There exists a suite of web scripts that present this data graphically.

2.5 Watchdog

A watchdog ensures that any crashed or hung process is restarted after short time.

3 Plugin Support

3.1 Concepts

The plugins communicate with the framework through functions of the UPFrame shared library. The library communicates with the Mgmt Process via a named pipe. On behalf of the plugin, the Mgmt Process switches to the next younger shared memory segment etc.

A TrafficShaper entity buffers data on behalf of a plugin. The advantage in using the Traffic Shaper is that the plugin is not affected by the burstyness of the traffic, the data is delivered to the plugin in a sustained data rate. This helps when resending the data over a network or when writing the data to a tape streamer etc.

The convenience of having a sustained data rate comes at the cost of a larger, more variant latency as the data has to be buffered. The amount of buffered data is configurable and can go as high as hundreds of megabytes of data.

Every plugin can define a maximum data rate that is not exceeded. Therefore for every plugin there exist its own TrafficShaper entity.

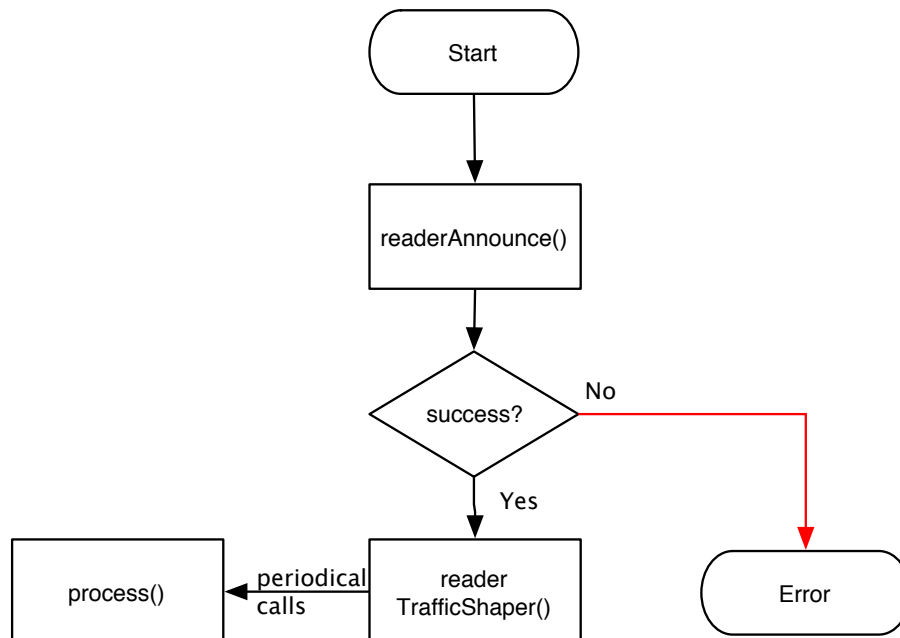
A problem arises when a plugin is (temporarily or all the time) slower in processing as there is data flowing into the buffer. As the buffer space is limited the Mgmt Process has to decide when to rob a segment from a plugin. The distance of a segment is meant as numbers of segments from the current segment to the newest one, the newest segment has always distance 1.

There are two possibilities for segments not to be (fully) processed by a plugin:

- The plugin finishes processing a segment. At the time of finishing, the distance of the segment is larger than `SoftLimit` (a value which can be set in the config file). The data just read from the segment was perfectly valid. The `readerAdvance` call now does not return the next younger segment but jumps to the segment with distance `SoftLimit`, the segments between the just finished segment and the `SoftLimit` segment are lost but the data read was valid. Either the plugin speeds up and the distance gets smaller or the plugin gets into a sampling mode where only some segments are read.
- If a plugin is extremely slow it can happen that the segment, which the plugin is currently reading, is taken out of the read list and gets in the free list. The segment will either be rewritten or trashed in the future. Until this happens the data in the segment is still valid, therefore the library checks for every access the validity of the data: in case the segment is removed or reused this validity test fails. The plugin has to advance to the segment with distance `SoftLimit` although the plugin did not read the whole segment. Not only the data between the current segment and the segment at distance `SoftLimit` but also part of the current segment is lost but the data read was perfectly valid.

Loss of data is indicated either as return value of the `readerGet()` or `readerAdvance()` call when using no TrafficShaper support or as parameter `loss` of the function `process()`.

3.2 TrafficShaper based



Writing a plugin which uses the TrafficShaper is very easy. The starting point is the TrafficShaper skeleton (see Appendix). In the skeleton the function `readerAnnounce` is called. Its parameters are:

- The path to the directory where the named pipes aka fifos reside
- How the plugin is restarted if the watchdog detects a crash
- How long the plugin can wait between two consecutive watchdog calls
- An empty watchdog handler to be filled by `readerAnnounce`
- The name of this plugin (if more than one plugin is concurrently installed in the framework they should have distinct names in order to be distinguishable)

When the plugin is known to the Mgmt Process it can start the TrafficShaper. The function `readerTrafficShaper` is called with

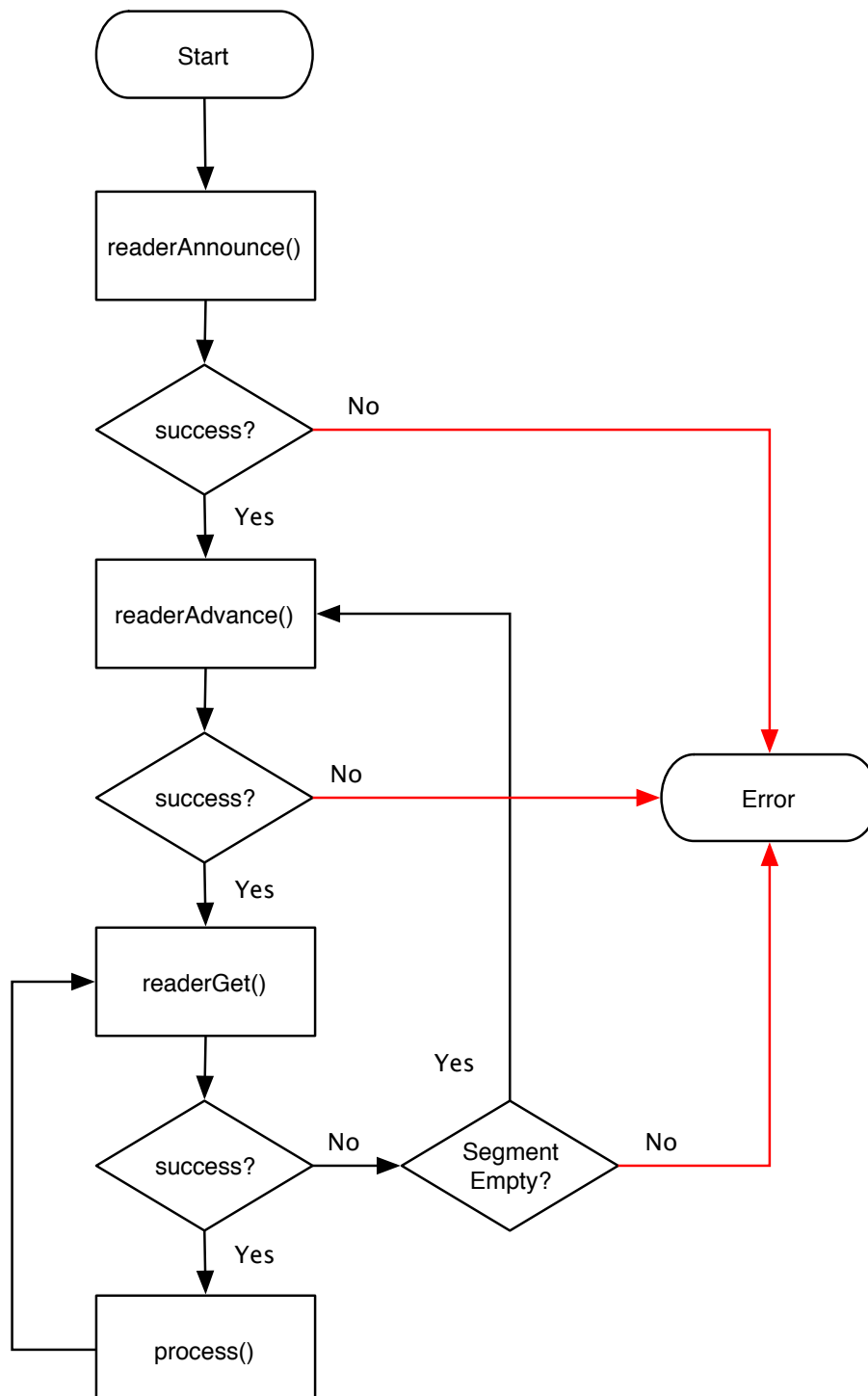
- The maximum bandwidth in Bytes/s
- The callback function which is called for every received UDP packet.

The callback function gets called in regular intervals. These intervals are calculated so that the output is as even as possible while never exceeding the maximum bandwidth.

The parameters of the callback are:

- The received packet
- An indicator if data was lost

3.3 Without TrafficShaper



Without the TrafficShaper a plugin gets more complicated but using the skeleton it is still fairly easy. The announce procedure is still the same. But now the plugin is responsible to advance to new segments and to get the data from the segment. Therefore after announcing the plugin has to call `readerAdvance()`. This function has no parameters. When the function is called directly after announcing the segment it advances to is the freshest. The plugin then calls `readerGet()`. Its parameters are:

- the address of a pointer later pointing to the packet
- a pointer pointing to an int var where the size of the memory region will be copied to

The variables the pointers are pointing to have to be set to NULL resp, 0. before the first call to `readerGet()`. `readerGet()` allocates memory and writes its address and size to the given variables. The received data is copied into the memory segment. This segment is enlarged on demand, therefore the allocated address and size have to be passed on every call.

Calling `readerGet()` is repeated until there's no more data in this segment. In this case the plugin has to call `readerAdvance()` to get a fresh segment.

For a new plugin the skeleton files are recommended as starting point.

4 Installation

4.1 Prerequisites

The UPFrame core needs the following prerequisites:

- gcc \geq version 3.3 (find out with `gcc -v`)
- glibc \geq version 2.3 (`ls -la /lib/libc.so.6`)
- make \geq version 3.8 (`make -v`)

The core components work with a 2.4 and 2.6 kernel if the kernel is compiled with the following option:

```
CONFIG_SYSVIPC=y (set per default)
```

The UPFrame web tools are tested against the following packages:

- php 4.3.4
- rrdtools 1.0.45
- php4-rrdtool-1.03
- perl 5.8.2

4.2 Compilation

The main makefile contain some items that have to be configured at compile time:

```
#####
#   COMPILE TIME OPTIONS   #
#####

# the framework will be installed here
INSTALLDIR = /usr/local/

# Maximum number of Client Plugins
MAXC = 25

# Maximum number of Receivers
MAXR = 20

# Number of filter coeffs
FILTER = 100

#####
# END COMPILE TIME OPTIONS #
#####
```

These values have to be set prior to compilation and cannot be changed at runtime.

- `INSTALLDIR` designates the directory where `make install` will install the framework.
- `MAXC` is the maximum number of concurrent plugins. When changing this number keep in mind that more plugins need more memory for managing the plugins (kept in shared memory) and that more plugins pose a higher load to the cpu.

- **MAXR** is the maximum number of receivers. For statistics the framework counts bytes and packets it receives from a certain host or subnet. Each receiver corresponds to such a host or network. **MAXR** is equivalent to the maximum number of different hosts or subnets the system can discriminate.
- **FILTER** The TrafficShaper uses a low pass filter to flatten out short bursts of data. The **FILTER** value determines how many filter steps this filter has. See the section about the filter for details.

After configuration the framework can be compiled with `make`. `make install` will then install the code in the designated directory.

4.3 Configuration

4.3.1 Directory Layout

The install directory now contains the folders `bin` `etc` `var` and `lib`. While `bin` holds the executable files, `lib` contains the shared libraries used by the system. Either `INSTALLDIR` was set to `/usr/local` so the system can look for the shared library in standard places or one has to show the system where the shared libraries reside. There exist two possibilities:

- Usage of `LD_LIBRARY_PATH`: The environment variable can be set to the path of the lib directory.
- Appending the `lib` directory into `/etc/ld.so.conf`. This is the more practical solution but needs root access on the machine.

4.3.2 Environment Variables

There exist two environment variables the system can be configured with:

- **ADMINPAGE** designates the key of the administrative shared memory segment (called adminpage). This (decimal) key can be freely chosen as long as it's not already used by some other package (use `ipcs` as user `root` to check). It is also one part used to distinguish between several instances of the framework running simultaneously besides the directory where the named pipes reside.
- **FIFOPATH** should point to a directory where the named pipes aka fifos are kept. Fifos are used as a means of communication between the several processes of the core system. As in **ADMINPAGE** this value should be set separately for every instance of the framework.

4.3.3 Config Files

Several aspects can be configured in the framework using the config files. There are sample files installed in the `etc` directory.

4.3.3.1 mgmt.cfg

`mgmt.cfg` is used to configure the memory manager of the framework:

Keyword	Default	Description
<code>segmentsize</code>	131072	size of a memory segment in Bytes
<code>fifoPath</code>	<code>var/fifo</code>	path to named pipes/fifos
<code>freeFifoSize</code>	80	size of free fifo
<code>freeFifoWarn</code>	15	free fifo warn level

readFifoSize	80	size of read fifo
readFifoWarn	20	read fifo warn level
freeBufferMin	160	minimum segments in free buffer
freeBufferMax	320	maximum segments in free buffer
freeBufferWarn	10	warn level of free buffer
readBufferSize	800	maximum segments in read buffer
readBufferWarn	50	read buffer warn level
softLimit	780	position of SoftLimit
trashTimeout	10	lifetime of a segment in trash in sec
trashBufferMax	20	maximum segments in trash
lowerlimit	10	limit between usage of filtercoeff1 and filtercoeff2
upperlimit	70	limit between usage of filtercoeff2 and filtercoeff3
filtercoeff1	20	filter coefficient applied to young values
filtercoeff2	5	filter coefficient applied to medium values
filtercoeff3	1	filter coefficient applied to old values

Sizes

The segment size, the fifo and buffer sizes are subject to tweak for a certain application. A small segment size lowers the latency of the buffer but the system load rises. The buffer and fifo sizes are given in number of segments. Therefore if the segment size is changed the buffer and fifo lengths have to be changed accordingly!

Care has to be taken so that freeFifo has always free segments available for the writer. If the free buffer size is small, the load of the machine rises because many shared memory segments have to be requested from the operating system but the memory footprint lowers. The position of the SoftLimit (see the corresponding section) should be near the end of the read buffer but the slowest plugin should be able to process a segment in the time it takes the writer to produce readBufferSize - SoftLimit segments. The buffers should be larger in size as the corresponding fifos.

Warnings

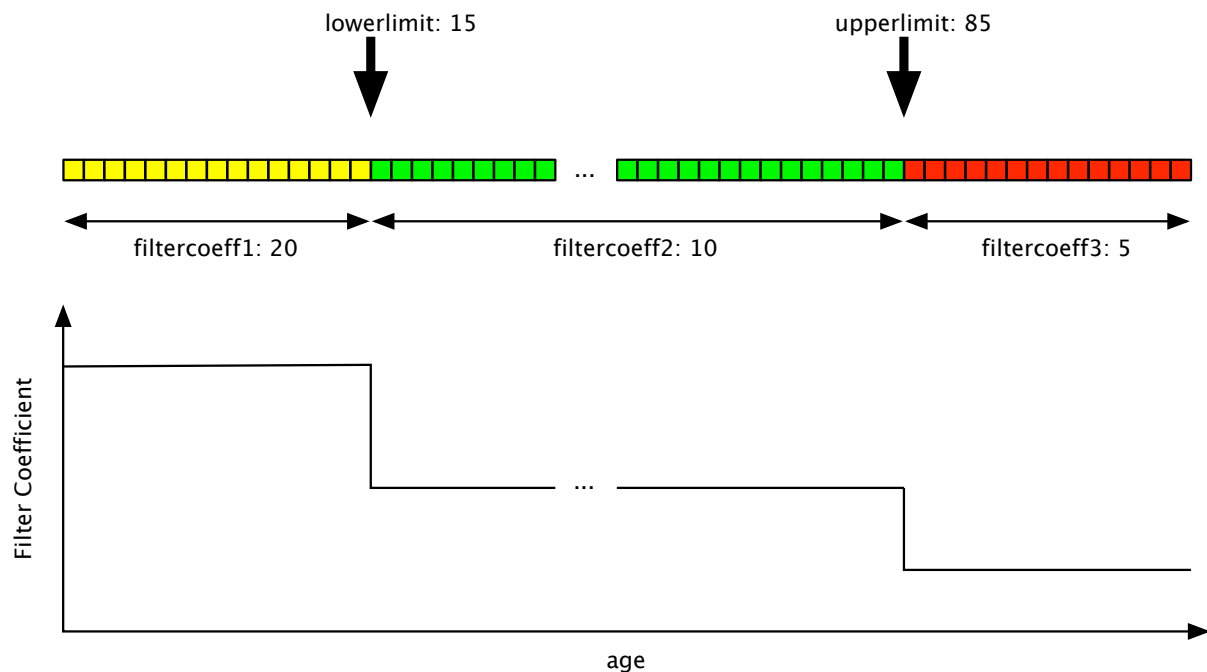
The framework guards some crucial values so they stay in safe bounds. An example for such a value is the actual fill level of the FreeFifo. When it drops below a value (freeFioWarn) in the statistics section a flag is raised and the number of occurred warnings is incremented. The statistics monitor then has to react (for example by sending email or sms).

SoftLimit

The meaning of the SoftLimit configuration variable is explained in the section about Plugin Concepts above.

Filter

The TrafficShaper needs to flatten out smaller bursts in the incoming traffic while adapting to high sustained load. One of the mechanisms is a FIR low pass filter. The filter coefficients can be configured as shown in the picture below.



4.3.3.2 writer.cfg

Keyword	Default	Description
port	19991	port where the senders send the UDP packets to
receiver	n/a	identifier of receiver, integer between 0 and MAXR
sender	0.0.0.0	IP of the sending host/net
mask	255.255.255.255	netmask of the sending host/net

Notes:

The receiver – sender – mask triple can be repeated. The maximum number of receivers (and the highest identifier) is the compile time option MAXR.

4.3.3.3 startpw.sh

startpw is an example start script that starts the core system and an example plugin, portwatch:

```
#!/bin/bash

export TOP=/usr/local/

export LD_LIBRARY_PATH="$TOP/lib"
export BIN="$TOP/bin"
export VAR="$TOP/var"
export ETC="$TOP/etc"
export FIFOPATH="$VAR/fifo"
export ADMINPAGE=3000

echo "starting mgmt"
```

```

$BIN/mgmt -c $ETC/mgmt.cfg -f $VAR/mgmt.log &

echo -n " portwatch"
$BIN/portwatch -a $ETC/pwaddr.cfg -c $ETC/pwports.cfg \
-f $VAR/portwatch.log &

echo -n " writer"
$BIN/writer -c $ETC/writer.cfg -f $VAR/writer.log &

echo " done."

```

It is recommended that the Mgmt Process is the first program started. Instead of using the environment variables FIFOPATH and ADMINPAGE one can use the corresponding command line switches `-a` and `-p`.

4.4 Contributions

As separate downloads there exist sample web scripts to present the data gathered by the statistics daemon `statd` as well as `portwatchd`.

4.4.1 Statistics Viewer

Several statistical parameters of the system can be logged and plotted using the RRDTool suite. Its values are the same as in the command line `stat` utility.

4.4.2 Portwatch

The Portwatch plugin is a supplied example plugin performing an analysis over the distribution of the ports in the ip packets crossing the Border Gateway Routers. It generates a summary over some of the more interesting ports (http, smtp, other privileged and unprivileged tcp ports etc.). It is possible to set some probes to observe some protocols or protocol ranges on demand. The plugin also calculates the top 30 of the used network protocols in tcp, udp and icmp. One can define a home network. The code then discriminates between incoming and outgoing traffic. Since the routers can cache the flow data up to some configurable time (for example 15min) there's a delay of 15 minutes in these analyzes because until then not the complete data is available. The web script is responsible to gather and present this data using the RRDTool.

4.4.3 Installation of the Web Scripts

The web front end scripts come in a tarball of their own. It contains a Makefile that has to be configured:

```

#####
#   COMPILE TIME OPTIONS   #
#####

# the web daemon scripts and the RRD
# databases will be installed here:
export SCRIPTDIR = /home/upframe

# the html pages and web applications

```

```

# will be installed here:

# documentroot of web server
export WWWTOP = /var/www/localhost/htdocs
# path below documentroot
export WWWDIR = /upframe

# the rrdcgi applications will be installed here
export RRDCGIINST = /var/www/localhost/upf_statd
export RRDCGIDIR = /upfcgi

# user that runs the daemon scripts
export OWNER = upframe

# user that runs the web server
export WEBGROUP = apache

#####
# END COMPILE TIME OPTIONS #
#####

```

After configuration the code can be installed with `make && make install`. Note that the Makefile installs scripts in the webserver directories and needs access to these. Depending on the webserver configuration it is necessary to install as root. The following example has to be adapted to the webserver configuration and should be added to the `apache2.conf` configuration file. It is valid for both presentation scripts `statd` and `portwatchd`.

```

ScriptAlias /upfcgi/ "/var/www/localhost/upf_statd/"
<Directory "/var/www/localhost/upf_statd/">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
    AuthType Basic
    AuthName statd
    AuthUserFile /var/www/localhost/upf_statd/.htpasswd
    Require valid-user
</Directory>

```

Of course the `.htpasswd` has to be set up accordingly.

Without authentication the config file reduces to:

```

ScriptAlias /upfcgi/ "/var/www/localhost/upf_statd/"
<Directory "/var/www/localhost/upf_statd/">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

```

4.4.4 Installation of the Portwatch Plugin

The Portwatch plugin is supplied in the base distribution. It uses two config files:

The first is `pwaddr.cfg`. The networks defined herein belong to the “local” network. Traffic from a network not defined to a network defined is counted as incoming.

The first value is the network address the second the (cidr) netmask.

```
10.0.0.0 8
192.168.0.0 24
```

The second designates the currently configured probes:

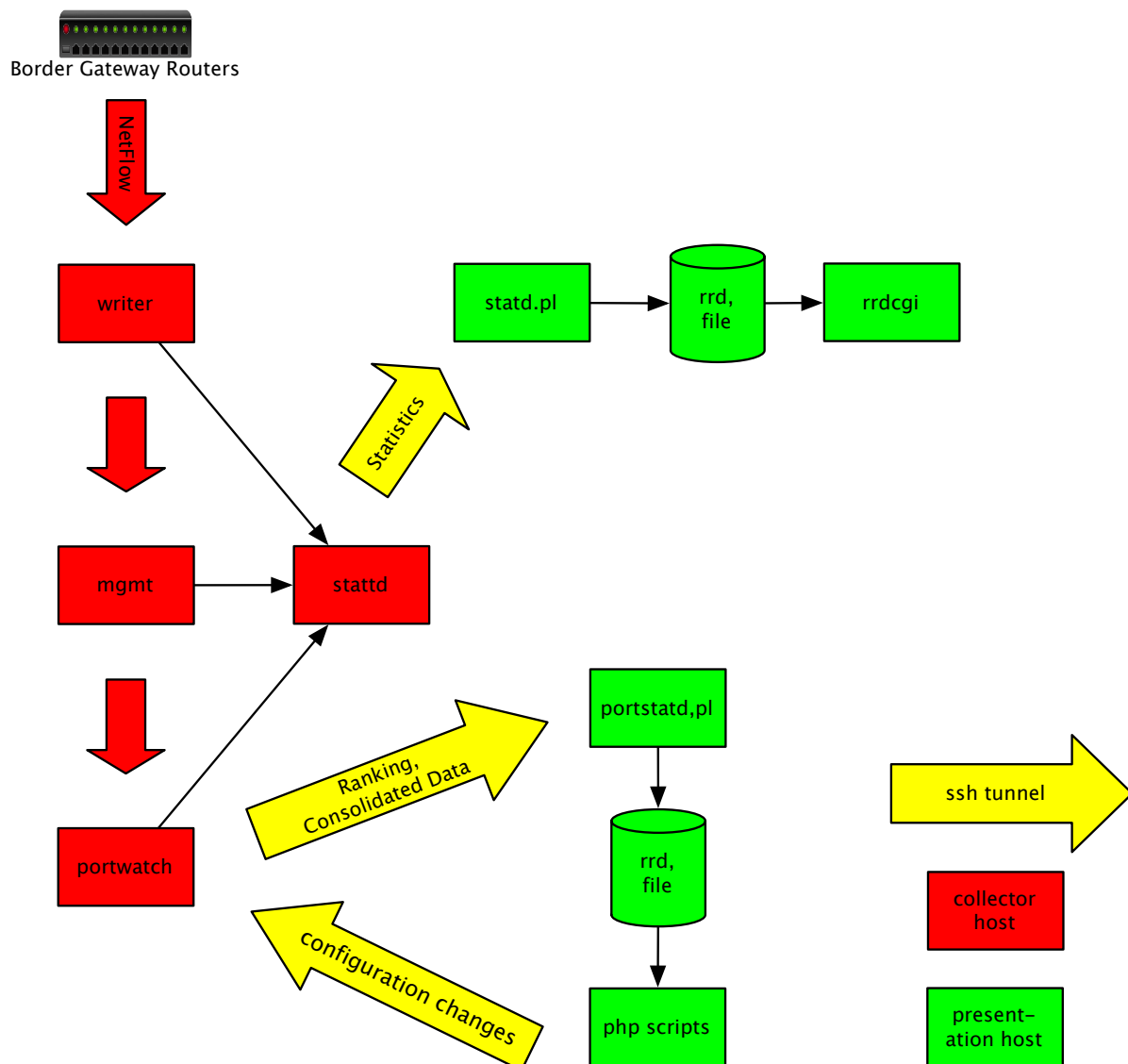
```
tcppriv 1 6 0 1023
tcpunpriv 2 6 1024 65536
ssh 3 6 22 22 irc 4 6 6666 6667
telnet 5 6 23 23
mydoom.F 6 6 1080 1080
```

This file can also be configured via the web interface. Its fields are:

```
name probe protocol lport hport
```

The probe named “name” with probe number “probe” catches data of protocol “protocol” (see /etc/protocols) when the port lies between the lower port “lport” and the higher port “hport” (borders inclusive).

5 Operation



This is the example setup when the core package and the Web Frontends are installed. The border gateway routers send the NetFlow data to the writer process. The writer process enqueues the received data in the buffer. The buffer is collected by mgmt and passed to portwatch.

Portwatch performs the analyzes and listens on two ports: on one port a connected program regularly receives a dump of the traffic counters of the various protocols as well as a dump of the top 30 data. The other port listens for connections from one of the php scripts: it allows the online configuration of the probes, the ports that are currently under surveillance.

on presentation host:

```
ssh -L20003:127.0.0.1:20003 -L20000:127.0.0.1:20000 \
-L20002:127.0.0.1:20002 collector.host -N
```

The mentioned ports are bound to localhost. Therefore one has to use some tunneling mechanism like an ssh tunnel in the example above.

5.1 Starting

The example setup as sketched above is started by running `etc/startpw.sh`. (See the section about the configuration files for an explanation)

The different programs of the framework have various command line switches, use `-h` to get a summary.

5.1.1 Multiple Instances

More than one framework can coexist on the same machine. A framework is discriminated from another by the `adminpage-id` and the `fifopath`, therefore every framework must have a unique `adminpage-id` and a unique `fifopath` (at least on the same machine).

The `fifopath` and the `adminpage` can be set in different ways:

- by using the `ADMINPAGE` and `FIFOPATH` environment variable
- by using the `-a` or `-p` command line switch

5.2 Statistics

The statistics tool `stat` creates various types of statistics:

5.2.1 Summary View -S

Field	Description
<code>smp</code>	Number of the current statistics sample
<code>FreeBF</code>	Level of Free Buffer
<code>FreeBW</code>	Number of Free Buffer Warnings
<code>ReadBF</code>	Level of Read Buffer
<code>ReadBW</code>	Number of Read Buffer Warnings
<code>FreeFLW</code>	Low Water Mark of Free Fifo
<code>FreeFHW</code>	High Water Mark of Free Fifo
<code>FreeFW</code>	Number of Free Fifo Warnings
<code>ReadFLW</code>	Low Water Mark of Read Fifo
<code>ReadFHW</code>	High Water Mark of Read Fifo
<code>ReadFW</code>	Number of Read Fifo Warnings
<code>Rcv</code>	Received Segments
<code>Snd</code>	Sent Segments
<code>bDisc</code>	Discarded Segments

5.2.2 Dataflow View -D

Field	Description
<code>smp</code>	Number of the sample
<code>Rcv</code>	Received Segments
<code>Snd</code>	Sent Segments
<code>Disc</code>	Discarded Segments
<code>Trashed</code>	Trashed Segments
<code>RP</code>	Number of attached reader plugins
<code>WP</code>	Number of attached write plugins (0 or 1)
<code>Mgmt</code>	1 if mgmt available

5.2.3 Memory View -M

Field	Description
smpl	Number of the sample
FreeBF	Level of Free Buffer
FreeBW	Number of Free Buffer Warnings
FreeBWS	Free Buffer Warn State (0 or 1)
ReadBF	Level of Read Buffer
ReadBW	Number of Read Buffer Warnings
ReadBWS	Free Buffer Warn State (0 or 1)
FreeFLW	Low Water Mark of Free Fifo
FreeFHW	High Water Mark of Free Fifo
FreeFW	Number of Free Fifo Warnings
FreeFWS	Free Fifo Warn State (0 or 1)
ReadFLW	Low Water Mark of Read Fifo
ReadFHW	High Water Mark of Read Fifo
ReadFW	Number of Read Fifo Warnings
ReadFWS	Read Fifo Warn State (0 or 1)

5.2.4 Receiver View -R

Field	Description
smpl	Number of the sample
Rec	Identifier of Receiver for following two fields
Packets	Packets from that source
Bytes	Bytes from that source

5.2.5 TrafficShaper View -T

The traffic shaper debug values are mainly for debugging reasons.

Field	Description
smpl	Number of the sample
Speed	time for sending current segment [usec/segment]
fcoeff	flush coefficient
ccoeff	corrective coefficient
fillr	buffer fill ratio
filla	averaged fillr
wtime	time for receiving current segment [usec/segment]
wtimea	averaged wtime
est	filtered wtime [usec/segment], estimation for equilibrium

5.3 Running the Watchdog

There is an optional watchdog support that checks if a component is still living. Every component under surveillance periodically refreshes a variable of the adminpage shared memory segment. If a component hangs or crashes it stops to refresh this variable. The watchdog checks for these variables and in case of the crash of one component the faulty component gets killed and restarted. A special case arises when `mgmt` is restarted. As the other processes have to register at `mgmt` they have to be informed to do so. This is done using a `SIGHUP` signal.

The watchdog support in the framework components is enabled by providing a watchdog restart command using the command line option `-w`. This restart command is a shell script that starts

the component using the correct command line switches (again containing the `-w` switch, for example:

```
startmgmt.sh:
```

```
#!/bin/sh
./bin/mgmt -w startmgmt.sh -a3000 -r
```

The `-r` switch of `mgmt` is special: it instructs the `mgmt` component to scan for lost shared memory segments while starting. Assuming that the `mgmt` component crashed or was killed by `watchdog` shared memory segments now lie in the system.

Shared memory segments can be free or can contain data:

- Unused shared memory segments are freed,
- Data in shared memory segments is not lost but can be processed by the plugins under government of the new Mgmt Process.

A plugin can benefit from `watchdog` support by providing the restart command and a timeout during the `readerAdvance()` call. The skeletons provided below already contain `watchdog` support and refresh the mentioned shared memory variable.

6 Guide to Plugin Implementation

6.1 Plugins without the TrafficShaper

Use the following skeleton to create a new plugin. The functions used are documented in the file `libread.h`.

```
#include "libread.h" // fetch the read functions

#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <stdlib.h>
#include <syslog.h>

#include "bufferpacket.h" // data types for payload
#include "udppacket.h"    // packets
#include "log.h"
#include "getenvs.h"      // environment variable handling

int myreconnect = 0;

/* default values for command line switches and environment variables */

char *restartCommand = NULL;
char *path = "/tmp/nerf";
int logfacility = LOG_TO_STDOUT;
char *logfile = NULL;
int timeout = 60; // watchdog timeout
struct sigaction oldsa;

/* prototypes */
inline int process(bufferPacketp_t packet, int result);
static void sighupHandler(int signal __attribute__((unused)));

/* we get sent a signal when the Mgmt Process exits. In this case
   we wait and try to reconnect.
*/
static void sighupHandler(int signal __attribute__((unused))) {
    oldsa.sa_handler(signal); // stack signal handlers...
    myreconnect = 1;
}

// process data
inline int process(bufferPacketp_t packet, int result) {

    // insert payload here

    return 0;
}
```

```
}

int main(int argc, char **argv) {
    int ret;
    struct sigaction sa;
    whandler_t whandler;
    bufferPacketp_t rec=NULL;
    struct timeval wait;
    unsigned int len;
    int loss;

    getEnvs(NULL, &path); // get environment variable for fifopath

    /* initialize logging */
    logwriteSetMedium(logfacility, logfile, "skeleton");
    logwrite(LOG_DEBUG, "skeleton started"); // and log

    while(1) { // a plugin normally just keeps running...

        printf("Announcing...");

        /* Try to announce. If there is no Mgmt Process, keep trying */
        while ((ret = readerAnnounce(path, restartCommand, timeout,
                                     &whandler, "skeleton")) == -2) {
            sleep(1);
        };

        // install signal handler
        sa.sa_handler = sighupHandler;
        sigemptyset(&sa.sa_mask);
        sa.sa_flags = 0;
        sigaction(SIGHUP, &sa, &oldsa);

        printf(" done.\n");

        /* while there's no data: */
        while ( (ret = readerAdvance()) != READOK) {

            if (restartCommand != NULL) {
                /* serve watchdog */
                watchdogSendHeartBeat(&whandler);
            }

            wait.tv_sec = 1;
            wait.tv_usec = 0;

            select(0, NULL, NULL, NULL, &wait);
        }

        // now we've finished preparing, start main loop.
```

```

while(1) {
    ret = readerGet(&rec, &len); // get bufferPacket

    if (restartCommand != NULL) {
        /* serve watchdog */
        watchdogSendHeartBeat(&whandler);
    }

    if (ret != READOK) { // error occurred during readerGet

        if (ret == READERR) {
            exit(1);
        }

        // in all other cases we've to advance to the next segment.
        // for this we need the Mgmt Process therefore we check if
        // we got the reconnect signal/

        if (myreconnect) {
            break; // finished reading block now go back for reannounce
        }

        if (ret == READSTALE) {
            loss = READSTALE; // data lost: block no more there
        }

        // else: READNODATA

        if ( (ret = readerAdvance()) == READSEQ) {
            // block loss detected in sequence number
            loss = READSEQ;
        } else if (ret == READERR) { // error in communication

            logwrite(LOG_DEBUG,
                    "communication error, sleeping 1 sec at %s, %d",
                    __FILE__, __LINE__);
            wait.tv_sec = 1;
            wait.tv_usec = 0;
            select(0, NULL, NULL, NULL, &wait);

        } else if (ret == READNOMGMT) { // error in communication, reannounce
            break; // go back to reannounce
        } else if (ret == READNODATA) { // no data
            wait.tv_sec = 0;
            wait.tv_usec = 100;
            select(0, NULL, NULL, NULL, &wait);
        }
    } else { // data is ok.
        process(rec, loss); // process read block
        loss = READOK;
    }
}

```

```

    }
  }
}

```

6.2 Plugins using the TrafficShaper

Using the TrafficShaper is even more easy: Use the following skeleton:

```

#include "libread.h"

#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>

#include "bufferpacket.h" // payload types
#include "udppacket.h"
#include "log.h" // for the logging
#include "libnetflowutil.h" // utility functions for processing netflow
#include "getenvsh.h" // environment variable handling

int ret;
bufferPacket_t *rec;
struct timespec ts;
char *infile;
char *restartCommand;
int timeout = 60;
int logfacility = LOG_TO_STDOUT;
char *logfile = NULL;
char *path = "/tmp/nerf";

// prototype
int process(bufferPacket_t packet, int result);

int process(bufferPacket_t packet, int result) {

    // insert function here

    return 0;
}

int main(int argc, char **argv) {

    int ret;

    whandler_t whandler;

    getEnvs(NULL, &path); // get environment variable FIFOPATH

```

```
if (logwriteSetMedium(logfacility, logfile, "skeleton") != 0) {
    fprintf(stderr, "could not open logfile: %s\n", strerror(errno));
    exit(1);
}

logwrite(LOG_DEBUG, "skeleton started");

while(1) {

    printf("Announcing...");

    while ((ret =
            readerAnnounce(path, restartCommand, timeout, &whandler,
                           "skeleton")) == ANNOUNCENOMGMT) {
        // wait until mgmt is available
        sleep(1);
    };

    if (ret == -1) {
        exit(-1); // error while announcing
    }

    printf(" done.\n");

#ifdef LIMIT
    /* this function will repeatedly call process and ideally never
       return. Here the traffic is limited to 1'000'000 Bytes/s in
       maximum
    */
    if (readerTrafficShaper(1000000, process) == READERR) {
#else
    /* no limiting here... */

    if (readerTrafficShaper(0, process) == READERR) {
#endif
        /* readerTrafficShaper only returns in case of an error. */
        logwrite(LOG_DEBUG, "TrafficShaper returned with error");
        exit(0);
    }
    printf("shaper exit\n");
}
}
```

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation’s software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too. When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author’s protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors’ reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone’s free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
 - b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
 - c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution,

a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

- c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by

public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy name of author
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

reade